Weakening Real-time Constraints for Embedded Control Systems

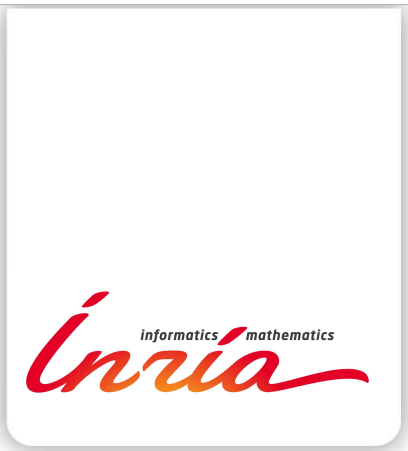
Patrick Jocelyn Andrianiaina, Daniel Simon, Alexandre Seuret, Jean-Michel Crayssac, Jean-Claude Laperche

Weakening Real-time Constraints for Embedded Control Systems

Patrick Jocelyn Andrianiaina , Daniel Simon, Alexandre Seuret, Jean-Michel Crayssac, Jean-Claude Laperche

RESEARCH REPORT
N° 7831
December 2011
Project-Teams NeCS

ISSN 0249-6399 ISRN INRIA/RR--7831--FR+ENG



Weakening Real-time Constraints for Embedded Control Systems

Patrick Jocelyn Andrianiaina* †, Daniel Simon†, Alexandre Seuret‡,
Jean-Michel Crayssac*, Jean-Claude Laperche*

Project-Teams NeCS

Research Report n° 7831 — December 2011 — 22 pages

Abstract: In this report, we propose to approach the problem of end-to-end performance achievement of real-time control loops in a global and system view. Conversely with the classical approach based on the separation of concerns between control and implementation, a coarser-grain view is investigated. The new approach is intended to be less architecture-dependent and able to provide the system with fault-tolerance abilities insuring robustness.

Key-words: Aerospace computer control, real-time, robustness, performance evaluation, fault tolerance, timing jitter.

* Airbus, Avionics & Simulation Products - EDYYAS, 31060 Toulouse Cedex 09
† INRIA Grenoble Rhône-Alpes, NeCS project-team, Inovallée Montbonnnot, 38334 St Ismier Cedex
‡ NeCS Team, Department of Automatic Control, GIPSA-Lab-CNRS UMR 5216, Grenoble, France

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Relâchement des contraintes temps-réel dans les systèmes de commande embarqués

Résumé : Dans ce rapport, nous proposons d'appréhender le problème de la performance d'une boucle de commande (feedback) exécutée par un système informatique temps-réel par une approche de bout en bout. Contrairement au point de vue traditionnel, reposant sur une stricte séparation de points de vue entre Automatique et Informatique, il est montré qu'une approche plus globale peut mieux prendre en compte les spécificités des commandes par feedback. L'implémentation de ces contrôleurs peut alors être réalisée de façon moins coûteuse tout en préservant les performances désirées.

Mots-clés : Contrôleurs numériques embarqués, temps-réel, robustesse, performance de commande, tolérance aux fautes, gigue.

Contents

1 Introduction

The development process of critical avionics products are done under strict safety regulations. These safety regulations include determinism and predictability of the systems' timing behavior. Indeed the overall approach is based on a separation of concerns between control design and implementation, e.g. Årzén et al. [2000] and Xia and Sun [2006].

On one hand, traditional control design considers constant sampling rates with equidistant samples (e.g. no jitter) and negligible, or fixed and known delays. On the other hand real-time scheduling theory has mainly focused on how to dimension resources for meeting deadlines (or equivalently, on the schedulability analysis for a given resource) Sha et al. [2004]. Therefore the computer science and real-time scheduling communities does their best to implement control tasks considering fixed periods and hard deadlines, and assuming that the WCET (Worst-Case Execution Times) is precisely known. This assumption has served the separation between control and scheduling designs, but lead to under utilization of CPU resources, and such approach faces both technical, economical, and industrial challenges.

One of the toughest challenges in the current approach is the determination of the WCET, needed, in order to correctly size the system. The tightness of the result is related to the predictability of the processing unit. The future generations of processors seem to go apart from the predictability and determinism objectives of the execution time. Processing speeds and performances grow up very fast thanks to accelerating but unpredictable mechanisms of new processors but it becomes very difficult to predict their effects on the execution time considered in the worst case. Nowadays, even if many attempts are proposed to give an upper bound of the WCET (e.g. Rochange

[2007]), both the traditional and current approaches are difficult to apply to modern processor generations and produce values which are pessimistic (Puschner and Schoeberl [2008]).

Anyway implementations purely based on WCET and hard deadlines considerations are by essence very conservative, leading for a large under-utilization of the computing and networking resources and consequently, leading to an oversizing of both electrical supplies, cooling systems and aircraft weight. The hard and costly way consists in building a highly deterministic system, from the hardware, operating system and communication protocols sides, so that the actual implementation parameters meet the ideal ones. To summarize the classical separation of concerns between control and computing leads to oversized and costly solutions (Xia and Sun [2006] and Årzén et al. [2000]).

Current real-time systems design methods and associated analysis tools do not provide a model flexible enough to fit well with control systems requirements, while classic control theory does not give advice on how to include resource and dependability constraints into the controller, both at the design and implementation stage. However, as far as *closed-loop* control systems are considered, more flexible solutions can be expected by exploiting the basic features of feedback loops, robustness w.r.t. modeling uncertainties, disturbance rejection and adaptiveness to various operative conditions.

In this position paper, we outline our current point of view about the issues and problems in safety and performance for computer controlled systems, the emphasis is given on coarse grain Execution Time considerations. That means that instead of approaching the problem at a fine-grain level, like using processors' instructions timings or memory access timings, we try to approach it with a end-to-end methodology, i.e., at the control system level. Indeed there already exist some results that support this point of view by formally demonstrating that a control system, although very critical, can be fault tolerant with respect to timing deviations from the theoretic timing pattern. The remainder of the paper exploits some of these results, it is organized as follows: Section 2 gives a detailed motivation of our approach with both the industrial and academic challenges to solve in terms of system design. Section 3 overviews existing approaches, results, methods, and tools which can be used in this framework. Section 4 formalizes our ideas on the problem. Finally, we give a conclusion in section 6.

2 Problem definition and Motivation

This section reviews some commonly known problematic concerning safety critical embedded systems and equipments.

2.1 System considerations and regulations

The design of critical system combines the domains of control systems, computer networks and real-time computing. Historically, tools for the design and analysis of such systems and related to each discipline have been designed and used with limited interaction. the increasing complexity of modern computers require more integrated methodologies, specifically suited to critical embedded systems. From a theoretical point of view, the main problem to be solved is the achievement of multi-objective goals (i.e. a mixture of stability, performance and dependability requirements).

Industrial sectors such as aerospace, avionics, nuclear, and automotive are considered as critical because a failure may have severe consequences. These systems have

to meet stringent requirements of dependability and safety but also requirements on performance and usability and so, are submitted to severe regulations. In this part, we roughly explain the ideas behind these regulations. From the perspective of avionics, there are requirements on safety that need to guarantee safety of the order of 10^{-9} in the probability of unwanted events per hour of use. At the higher level of an aircraft for instance, these guarantees are achieved through strategies like redundancy (See Goupil and Marcos [2010]and Bertrand et al. [2009]). However, the above-cited works do not consider the system at equipment level, where, things happen quite differently.

Software and hardware designers have even more stringent requirements to meet in order to guarantee determinism and to achieve certifications of their products. These requirements lead equipment designers and manufacturers to set determinism and predictability as a priority and not take into account fault tolerance. This approach is known to be very conservative because apart from being long and difficult in implementation, designers and manufacturers of equipments must consider taking some spare safety margins, especially in timing analysis, to guarantee required timing objectives. This can be seen in the software/hardware development, during timing tests analysis and also in general equipment where redundancy is implemented as dissemblance or dissimilarity, meaning that the same functional software or hardware implemented twice or even more times differently to avoid common mode failures. Some well-known redundancy architectures are mainly used in practice such as the triplex voting or COM/MON (command and monitor) architecture. Anyhow, compliance with these requirements is crucial to obtain certification that allows the use of the equipments.

2.2 Importance of fault tolerance

The above reflection lets us figure out that safety critical equipments are rigid and completely conservative. Our idea is to spread fault tolerance ability through end-to- end component of the functional system (to be precise, for instance a fault tolerance ability to each control loop composing a real-time control system). In fact, nowadays, a digital controller is designed assuming a fixed sampling period T, and possibly assuming a fixed computational delay τ. These simplistic assumptions are seldom met in the target equipments. The controller will suffer from time-varying latencies (commonly called Jitters according to Buttazzo and Cervin [2007]) induced by pre-emption, speculative execution, cache memory misses, pipelining of code, etc (see Cervin [2003]). Most control systems, except in the case of failure due to hardware or software components, usually run with nominal behavior. However, even in the nominal modes, neither the process nor the execution resources parameters are ever perfectly known or possible to model. A very conservative way to cope with this problem consist in allocating system resources to satisfy the worst case. This results in the execution resources over-provisioned and thus wasted. From the control viewpoint, specific deficiencies to consider include poorly predictable timing deviations, jitters and data-loss (or data obsolescence in the sense of arrival be-lateness). We believe that our idea, combined with usual safety approaches such as redundancy at higher level, for instance at the aircraft level, as explained in Goupil and Marcos [2010], will permit us to be less conservative in the implementation and loosen requirements towards system development while achieving at least the same dependability/performance objectives. Following Shin et al. [1985], *what is called for is a procedure for specifying and evaluating a controller dependability range, performance, enabling systematic application and providing objective results that lend themselves to formal validation.*

2.3 Robustness

Designing and implementing systems are limited at first sight to satisfying requirements and specifications. But then, the performance level of the product varies as a function of noise and external perturbations. A product can be considered robust as long as its intended response is not altered (or barely altered) by non controlled perturbations and modeling uncertainties. An optimized product that only works in very particular known conditions is not robust. The objective of robust design is to optimize product performance along with minimizing sensibility to perturbations and uncertainties. Robustness is (and must be) a general concern that grows with system complexity.

For instance, it is known that small task core execution time modifications in systems with complex performance dependencies can have drastic non-intuitive effects on the overall system performance, and might lead to constraint violations. Hamann et al. [2006] claims that robustness evaluation using simulation is a tedious tasks and practically impossible for the reason that simulation models do not support many of the possible property changes (for instance: increased processor execution times or modified communication volumes). The same paper then demonstrates that the "naive Approach" (section 3.1) is not satisfactory and formal models are more appropriate. The authors propose an interesting formal approach to robustness of embedded real-time systems with definition of robustness metrics. In our work, the robustness we would like to refer to, is the one toward execution times of control tasks. In control system, dealing with closed-loop controllers may give the advantage of the robustness and adaptability of such systems. Robustness in control usually deals with the plant's parameters uncertainties, but in the present case the insensitivity or adaptability w.r.t. timing deviations from the theoretical pattern, such as jitter or deadlines miss, must be also investigated. For linear systems robustness can be quantified using phase margins, delay margins and module margins. It appears that a phase margin (when it exists) implies a delay margin (i.e. the maximum unmodeled constant extra delay that can be suffered before instability) and certainly a jitter margin, which is more difficult to quantify (Cervin et al. [2004]) but which can be experimentally shown (Cervin [2000]). The interesting point is that a feedback control system which is robust w.r.t. the plants parameters uncertainties is also robust, to some extent, w.r.t. timing deviations. Hence a feedback control system is not as hard as it is often considered in the literature, but should be better considered as "weakly hard", i.e. able to tolerate a predefined amount of timing deviations before leaving their specified allowed performance set (Bernat et al. [2001]).

2.4 Jitter and sampling equidistance

Controlling a continuous plant with a discrete digital system inevitably introduces timing distortions. In particular, it becomes necessary to sample and convert the sensors measurements to binary data, and conversely to convert them back to physically related values and hold the control signals to actuators. The sampling theory and the z transform became the standard tools for digital control systems analysis and design. A smart property of the z transform is that it keeps the linearity of the system through the sampling process. As the underlying assumption behind the z transform is equidistant sampling, periodic sampling became the standard for the design and implementation of digital controllers.

Note that, at the infancy of digital control, where computing power was weak and memory was expensive, it was important to minimize the controllers' complexity and

needed operating power. It is not obvious that the periodic sampling assumption is always the best choice : for example, Dorf et al. [1962] show that adaptive sampling, where the sampling frequency is changed according to the value of the derivative of the error signal, can be more effective than equidistant sampling in terms of the number of computed samples (but possibly not in term of disturbance rejection Smith [1971]). Hsia [1972] and Hsia [1974] provide a summary of these efforts. However, due to the constantly increasing power and decreasing costs of computing, interest in sampling adaptability and the related computing power savings has progressively vanished, while the linearity preservation property of equidistant sampling has helped it to remain the indisputable standard for years.

Therefore the commonly used assumption in digital control design is the steadiness of the sampling clock and the equidistance of the sensors samples and of the control actions. Implementing this assumption can be costly and even counter-productive. For example it is generally considered that jitter degrades the control performance, which may be true out of practical implementation issue. However achieving null (or negligible) jitter induces architectural constraints which may jeopardize the expected gains. For instance the case studies examined in Buttazzo and Cervin [2007] show that an effective method to minimize the output control jitter systematically delays the output delivery at the end of the control period : however, this method also introduces a systematic one period input/output latency, and therefore most often provides the worst possible control performance among the set of considered strategies.

Finally control and scheduling co-design approaches, e.g. as introduced in Eker et al. [2000] and Cervin et al. [2002], consider control schemes where the real-time scheduler is also an object that can be controlled, and where varying control clocks become one of the control inputs of the system.

2.5 Equipment Safety: Some sources of problems

From equipment view, problems are numerous. On one hand, the new generation of processing resources (processors, micro-controllers) represent a challenging problem for safety critical embedded systems designers from the point of view of complexity, timing and performance guarantee . In fact, the new mechanisms of acceleration in these component increases unpredictability of execution timing. On the other hand, embedded software and hardware have to meet severe timing requirements. One very important need is the ability of the computing unit to guarantee treatment resource availability. Nowadays, WCET evaluation is used to satisfy this need although this evaluation is quite hard to achieve. This difficulty, combined with the problem addressed by new processors in terms of predictability, becomes a difficult challenge for critical system design.

Nowadays, the best methods to compute out a fairly correct WCET evaluation are component-dependent and mostly hardware-dependent. That means, the execution time is different from a given processor to another one. Roughly, any time there is a need to change component, the analysis must be done from scratch. The proposed tools or methods are severely flawed by this dependence. So we also need to aim for a solution which is architecture- independent.

2.6 WCET calculation problems

The methods used to evaluate the WCET of a task are divided into three main classes (Kirner and Puschner [2005]). There are dynamics methods based on measurements

or on probability execution time patterns (see Deverge and Puaut [2005],Kirner et al. [2000], Bernat et al. [2002]), static methods based on the analysis of the code and/or the executable software (see Burguière and Rochange [2006],Souyris et al. [2005]), and hybrid methods based on the combination of both methods (see Lisper [2003]). However, these methods impose some limiting assumptions. If we brush these assumptions up, they can set be on the linearity of the code which means the prohibition from using some kinds of variables types or some control instructions, they can also be prohibition from using the cache memories of the processors. The reason is to get rid of difficulty to produce deterministic and predictable software that is easy to analyze in terms of WCET. During the timing analysis, designers still take some safety margins in order to insure that their WCET evaluation is really safe. According to Louise [2002], the enhancements brought to new generations of processors increases the uncertainty and decrease determinism and predictability of the execution timings of instructions. In fact, the execution time of a given instruction, when run on these processors, is no longer constant but variable. That means, using new processor generations doom designers to take even longer spare margins penalizing the system on resource usage. An attempt to approach the problem using QoS (Quality of Service) measurements is also proposed in Abdelzaher et al. [1997]. The advantage of this approach is to adapt online the Quality of Service of the system and then deal with unanticipated overloads and failures. However, QoS is a scalar relating the quality of the system at a given time and then poses all the problems we will relate in Section 3.2

3 Methods and tools

Since many years, number of attempts were proposed to solve the problems of Jitters in embedded systems. We can classify these works in terms of field of interests. Some solutions are purely in the scope of computer science. These works tend to solve more of the problems of conservatism, architecture dependence, new processor usage, or assumption taking in the methods of evaluating the WCET of a program. Other works propose safety considerations very early in the development process.. These works propose to integrate WCET considerations as early as in the modeling process of the system (see Kirner et al. [2000]). Last but not least, there are works that propose a system approach as we have described above, that is in terms of control systems and end-to-end blocks. A must-read paper on the field is Henzinger and Sifakis [2006] which gives an overview on the constraints, the need, the trends and industrial practices concerning embedded systems design methods. This paper shows that an embedded system requires a holistic approach that integrates essential paradigm from other fields (particularly from hardware and software design). As we stated formerly, we will focus on holistic methods taking into consideration control aspects of the problem.

Remark: The challenge and difficulty resides mainly in the fact that a system submitted to jitters becomes infinite-dimensional (Simon et al. [2009]) and also an incomplete knowledge of the relationship between control performance and control tasks scheduling parameters.

3.1 Naive approach

A naive strategy to solve the problem would be to take a commonly known control system and to apply the proposed ideas on it. In our experiences, we used the TRUETIME control system example shown in figure 1 (composed of a plant and a PID controller).

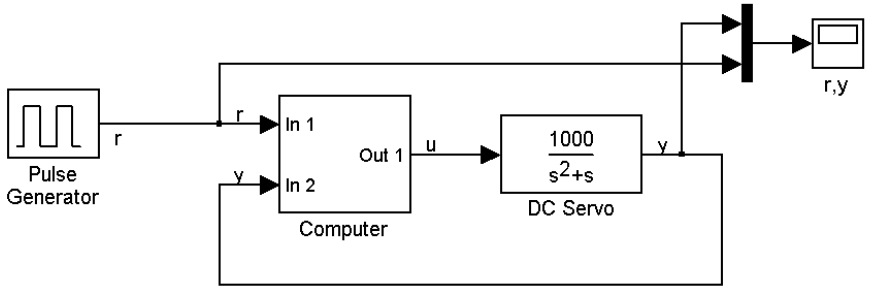


Figure 1: A truetime model for overrun simulation

The objective is to apply some fault pattern or scenarii on the control system and see how it behaves. Among the many scenarii applied, we have applied a random distribution of deadline miss pattern (meaning that a deadline miss happens randomly during execution as shown in the legend of figure 2

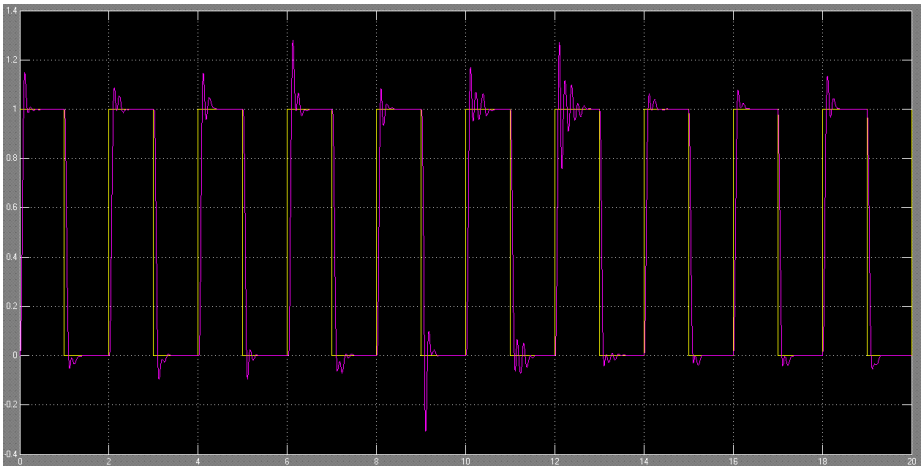


Figure 2: T=4 ms+rand(2ms)⇔WCET=5 ms: for 20 seconds

This approach shows us that control system can be fault tolerant in opposition to current beliefs. Ben Gaïd et al. [2008] gives a number of references related to this fact and the introduction of new task models in adequation.

This kind of experiments and dedicated simulation tools obviously fail to neither formally identify the timing faults patterns which can be born by the control system while keeping its state and behavior inside the specified bounds nor to formally prove any stability condition. However they are useful to provide some design directions and evaluate the sensitivity of a given system w.r.t. the control tasks scheduling parameters. For instance it can be used to evaluate overruns handling strategies such as those listed in Cervin [2005]: among various handlers, an overrun detection may lead to abort the current task and to cleanly start a fresh instance, or to finish the running one over the deadline and skip the next instance, with various consequences on the system's behavior.

3.2 Performance indices oriented solutions

According to what is called for, the need of conventional and common metrics is important. Generally, these involve representing the computer controller as a vector made up of n elements such that each element represent safety, reliability, availability survivability, etc. The problem occurs when it comes to compare two vectors. One solution is to assign some weights to each component of the vector. However, this approach is subjective and so does not meet our requirements. The same drawbacks happens to solutions proposed by Cervin [2003] ,Cervin et al. [2006], Cervin et al. [2006] and Seto et al. [1996]. Their solutions propose to associate the control system with a scalar performance index. The approach uses well-known optimal control theory results that associate the quality of control with a number called " Performance index". Given the control systems (1), The system performance is defined formally as in (2).

$$\begin{cases} x_d(t_{k+1}) = \phi\, x_d(t_k) + \Gamma\, u(t_k) \\ y(t_k) = C_d\, x_d(t_k) + D_d\, u(t_k) \end{cases} \tag{1}$$

$$J_d = S(x_d(t_k), t_k) + \sum_{k=0}^{N-1} L(x_d(t_k), u(t_k), t_k)dt \tag{2}$$

S(.) and L(.) are functionals depending on the system.
Remark: In Jitterbug (see Cervin and Lincoln [2003]), this performance index has the following form (3)for discrete-time systems:

$$J_d = \lim_{T\to\infty} \frac{1}{N\delta} \sum_{k=0}^{N-1} [x^T(k\delta)\, Q\, x(k\delta)] \tag{3}$$

where δ is defined as the unit time-grain.
However, in terms of practical application, we think that the solution is not satisfactory for highly critical domains such as avionics. The reasons is that these works propose a mathematical demonstration of the relationship between control system performance index and the control task frequencies but there is no talk about the specification of some important parameters of this approach such as the detailed specification of the cost function, the choice of the cost matrix. It is interesting to know that this approach has been implemented by a tool called Jitterbug (see Cervin and Lincoln [2003]) and when simulating with the tool, these problems said above show up. In some parallel research, we are currently attempting to improve this approach by applying some methods of choice of the weighting cost matrix as explained in Alden and Crusca [1992].

3.3 Adaptive sampling methods

To start this part, we illustrate the benefit that can be achieved by optimizing the sampling period of a controller, *table 1* shows a result from Seto et al. [1996] where the controller is a multi-rate controller composed of 5 tasks each one having its period and percentage processor load.

This set of task will never be schedulable because the processor usage is greater than 100%. The method proposed consists of using the performance index defined by (2) to derive the optimal sampling rate of the tasks set while keeping schedulability at its optimum. This solution is interesting but it does not consider the simplest notion of robustness. An on-line solution, that means dynamic reallocation, proposed by Shin and Meissner [1999] still uses an approach performance index but combined with

Table 1: Data for tasks in multi-rate computer controller

	C_i	F_i	Utilization(%)
Unit 1	10	30	30
Unit 2	15	20	30
Unit 3	20	20	40
Unit 4	25	10	25
Unit 5	30	10	30

some heuristic reallocation algorithm that permit to adapt the control system's period in discrete increments. The authors introduce then the notion of allowed period or frequency.

Beside methods leading to statically compute the values of fixed control tasks period through some optimization process, radically new methodologies has been introduced during the last decade. Among others, a remarkable milestone is (Eker et al. [2000]) which formalizes the sharing of a CPU resource between several control activities as an optimal control problem. Several plants are controlled using LQ controllers aiming at minimizing their respective control costs

$$J(h) = \frac{1}{h}\int_0^h \begin{pmatrix} x_c(t) \\ u(t) \end{pmatrix}^T Q \begin{pmatrix} x_c(t) \\ u(t) \end{pmatrix} dt \tag{4}$$

J(h) is the overall scheduling cost over the scheduling period.

The objective consists in minimizing a control cost over all the control tasks

$$\min_{h_i} J = \sum_{i=1}^{n} J_i(h_i) \tag{5}$$

under constraint

$$\sum_{i=1}^{n} C_i/h_i \leq U_d \tag{6}$$

where C_i is the execution time of control task i and U_d the set point for the desired CPU load and in which h_i is the control period of each control task.

A closed-form controller of this problem is given by:

$$h(k+1) = f(C_i(k), U_d(k)) \tag{7}$$

This form can be found but is too complex to be implemented in real-time systems. However it must be highlighted that here the control interval becomes one of the control signals, hence it is no longer considered as a constant. Therefore it is understood that the scheduling parameters of a real-time controller (e.g. the sampling intervals, priorities, deadlines, CPU speed,...) can be also objects that can be controlled via a *feedback scheduler*. Several approaches and case studies has been developed following this principle, to solve various specific control/scheduling co-design problems under restricted assumptions.

Among others a robust approach is provided by the H∞/LPV (Linear Parameter Variant) approach which proposes a robust on-line solution (Robert et al. [2010]). The approach considers the sampling interval as a varying parameter of the system to get a H∞ based design, where the requested control performance also varies according to the

expected sampling rate. The advantage is that the sampling period may vary at any time and speed inside predefined bounds while preserving the system's stability, and still have a coherent mathematical model to work with. This fact is important because with such a mathematical model, it is possible to demonstrate results theoretically, and then experimentally, to finally tackle certification objectives. According to our inquiries, some enhancements would be provided in terms of fault modeling. In fact, as in the Jitterbug tool, timing faults could be modeled as a Markov chain through probability density associated with a LPV model and robust control synthesis. This kind of fault model could be used to have a pattern of fault as an input of the model (taken from a processor characteristics' book or probabilistic experimentation for instance) and as output the behavior of the control system.

4 Control objectives and real-time constraints

Consider the system of figure 1. Among solutions, we can list the approach described in Section 3.2 concerning *performance indices*, the adaptation methods either on-line or offline cited in Section 3.3. An enhancement would be to specify rigorously the term "Performance Index" and " Quality of Service" and the inferred service. In fact, there are some domains where QoS is required and some others where determinism is demanded to guarantee safety. However, in the context of a real-time control system, these notions are not so far away from each other because performance naturally depends on the latency (i.e the interval between the sampling of order and the actuation) of the control task.

This considerations permits the specification of a required level of performance to insure safety and a certain degree of determinism. Only under those terms, may we use the notion of performance indices.

The problem resides in the fact that when the timing deviation is variable from sampling to sampling, classical control theory does not propose any solution to model the dynamics of such systems. In fact, when submitted to jitters, the resulting system becomes infinite-dimensional (Simon et al. [2009]) and moreover, there is no information on the relationship between control performance (such as performance index for instance) and the scheduling parameters.

An advance on the relationship problematic has been proposed by expressing the performance of the controller and the plant on the basis of a single variable: *the controller response time* (Shin et al. [1985]). This term is defined clearly as the interval of time between the initiation of the controller job (that means sampling the analogical order) and the actuation of the plant.

This will permit objective measure and estimation based on realistic facts such as the probability distribution of the controller response time and the controller's total capabilities. So this measure is far more reliable and effective that others in use.

One step forward would be a method that is able to formally bind, using classical control theories, the stability domains depending on the applied fault patterns. A related reflexion can be found with the notion of "Allowed State Space" and "hard deadlines for control systems" presented by Shin and Kim [1992] and Shin et al. [1985].

The notion of hard deadlines is then defined for control systems, as the critical value of the computation-time jitter beyond which the system leaves its allowed state-space and then becomes unstable. It is important to remark that the computation-time inferred here should be understood in the sense of computer-controller calculating the control signals. The whole feedback delay is assumed to result from the computation-

time jitter because other delays elements can be readily dealt with by classical delay theories. The term *hard deadline* is not *necessarily* associated to the sampling interval of the controller (neither is it to the task period of the control task). The hard deadline is instead derived from the controlled plant. The condition of asymptotic stability of the plant and making it stay within its allowed state space - which must hold to avoid failure- can be used to derive the hard deadline. Studies on *derivation of hard deadlines*, in terms of control system, show that a control system may tolerate up to many sampling intervals of jitters without going unstable and that missing one single task deadline may not be lethal for the system. Hard deadlines can be obtained by iteratively testing the necessary conditions of stability and state residence within the allowed state space running through what we introduced as the "admissible parameters space". The admissible parameters space is the set composed of all possible jitter (or delay) value added to the sampling interval value (eventually multiplied by the number of sampling interval) suffering the delay.

Allowed state space is, by definition, a set of state that the system must not leave during its execution in order to avoid catastrophic failure (figure 3)

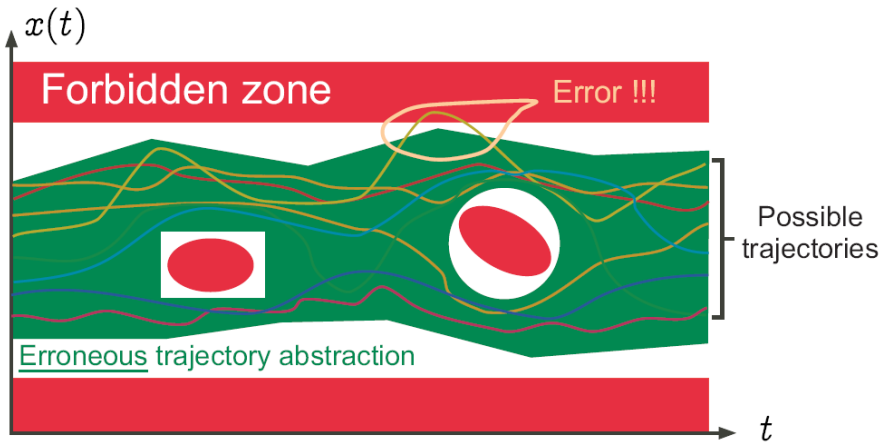


Figure 3: A case of failure: state out of allowed space

A practical example is a situation in which a landing aircraft flying upside down is considered as outside of its state space. The allowed state space is derived from given input and some state constraints. It can also be used to calculate the hard deadline whereas the knowledge of hard deadlines (in control systems point of view) allows derivation of the relationship between performance and the controller response time.

Some experiences and examples presented in Shin and Kim [1992] show that hard deadlines can also be derived from stability analysis and thus can prove that the system does not leave its allowed state space.

All requirements being gathered, this approach fits a formal mathematical method allowing qualitative analysis.

In this approach, the characteristics of transient computer failures are taken into account. The notion of "allowed state space" can be compared to a field of formal verification in computer science: "Abstract interpretation". It formalizes the idea that at some level of abstraction, and based on some demonstrated result, we can allow some mastered uncertainties within the execution of a control system and still demon-

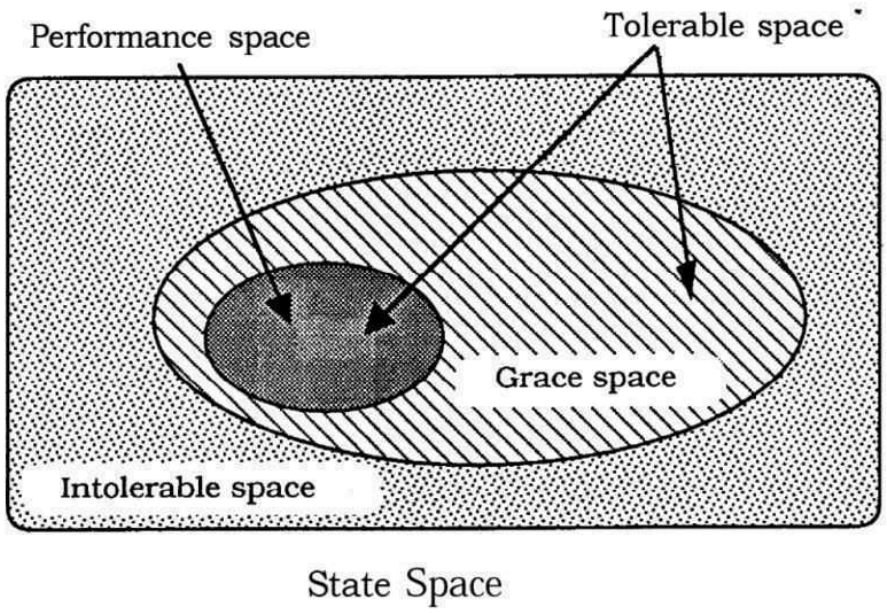


Figure 4: Allowed State Space

strate safety and performance. The advantage is to allow the use of classical analysis methods.

If we can draw a bound made up of some sets of state space, the problem greatly decreases in difficulty because, we only have to prove some properties on the boundedness of our initial model and also some other properties of safety and stability on the bounds of the sets. Supposing we have demonstrated our *state space envelop*, another step of this solution is to find an estimation of the system according to the (m,k)- firm formalism or according the weakly hard real-time approach (see Ben Gaïd et al. [2008] and Bernat et al. [2001]. This approach leads into finding the number of sampling period where the system faces deadline violations without going unstable.

This notion permits us to define some new notions of deadlines sorted by inverse criticality such as:

- performance deadlines
- tolerable deadlines
- the usual Hard deadlines

where each new notion of deadline is associated to a state subset, each one, included in the allowed state space.

However, there are still some difficulties that need to be mentioned. According to Shin et al. [1985], the method can be expensive in terms of computation resources in implementation, especially if a closed form solution of the "hard deadline" cannot be found. The objectivity of the performance measure must rely on a natural performance index of the plant (and this is relatively subjective). Last but not least, all these approaches are valid for Linear Time Invariant systems. Among our undergoing activities, we also seek for the way to adapt the idea for non-linear systems by using robustness to cope with non-linearizable parameters.

5 Weakened control tasks execution patterns

5.1 The WCET based scheduling case

Currently many control systems, e.g. flight control, braking control systems, are considered to be hard real-time, i.e. it is most often assumed at design time that control tasks must be executed strictly periodically. Therefore control tasks executions are bounded to fixed time-slots, and deadline misses or jitter are forbidden. It is assumed that any deviation from the ideal timing pattern inevitably leads to a failure of the system. The implementation of such control tasks relies on a safe evaluation of the WCET of each task, which is used to dimension the size of the time slot allocated for the execution of the control tasks. The execution schedule of a control task is depicted on Figure 5.

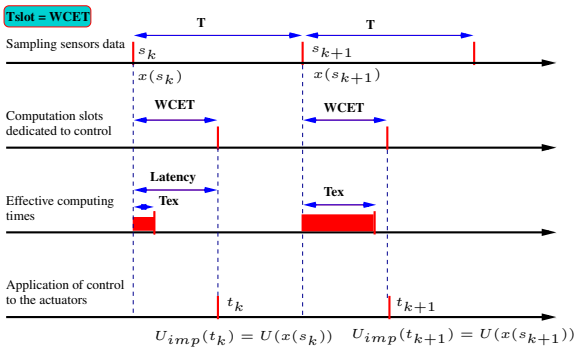


Figure 5: WCET based control task execution pattern

A given task execution is strictly periodic, i.e. a time slot $T_{slot} = WCET$ is allocated to the task execution. It is triggered at a period $s_k - s_{k-1} = T$ by the occurrence of measurements $x(s_k)$ at time s_k. The controller computation takes a time T_{ex} which is always smaller than the $WCET$. To avoid jitter, the control signal $U(x(s_k))$ is applied to the actuators at the end of the slot, i.e. at time $s_k + WCET$:

$$\forall t \in [s_k + WCET, s_{k+1} + WCET[, \quad U = U(x(s_k)).$$

Therefore, it is a periodic control system, with constant period T, subject to a constant delay $T_{slot} = WCET$. This implementation fits with the hard real-time assumption, and should be applied when the controller is really hard, e.g. if it is a Finite State Machine which may fail in an unpredicted state in case of deadline miss and interrupted transition.

However, as the time slots are allocated based on the WCET of the control tasks, the computations always finish before the end of the slot. Therefore, a fraction of the computing power is unused. The wasted computing power is all the more important as the WCET is far from the average value of the observed execution time T_{ex}, as it is the case with modern powerful processors using cache and pipelines. Therefore the amount of wasted computing power is expected to increase, leading to costly oversizing of embedded computers, power supplies and cooling systems.

5.2 Weakened scheduling scenarii proposals

As already observed and reported (e.g. Cervin et al. [2002]), it is likely that a robust feedback control systems can keep stability despite timing disturbances such as jitter and occasional data loss, and overrun handlers to process deadlines misses have been proposed, e.g. **Skip**, **Abort** and **Queue** as in Cervin [2005]. We propose here several patterns for control tasks executions under weakened real-time constraints. All these scheduling scenarii rely on the allocation of periodic time slots smaller than the WCET needed to compute a given feedback control task (Figure 5.2).

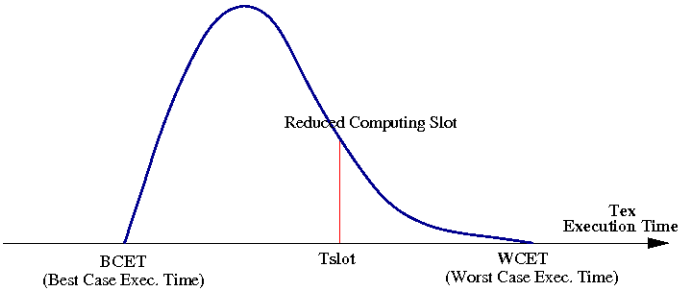


Figure 6: Reduced computing slot

To keep as close as possible to current practice, the control tasks triggering is assumed to be fired periodically by the occurrence of equidistant sensors inputs. Reducing the time slot below the WCET of the task allows for reducing the number CPU cycles wasted at the price of occasional deadlines misses. However, it also induces potential reductions of both the control period and of the I/O latency, leading to performance improvements. It is expected that the disturbances of the control loops induced by the timing faults can be compensated by the gains due to delays and period reduction. The proofs of stability are not given here, they are based on robustness arguments given in Seuret [2011] and exploited in Andrianiaina et al. [2011] for the first two cases described below.

To improve the average efficiency of embedded computers while preserving the control stability and performance, and relying on the robustness of feedback control laws, it is proposed to weaken the usual real-time constraints according to the following scheduling patterns :

1 Abort The sensors data are still expected to occur at a fixed period T, and their occurrence trigger the control tasks. The time slot allocated to a given task $T_{slot} < WCET$ is now smaller than the worst case. As usual the control signal is sent to the actuators at the end of the slot, i.e. $U(x(s_k))$ is sent at time $s_k + T_{slot}$, and the delay is equal to T_{slot}

$$\forall t \in [s_k + T_{slot}, s_{k+1} + T_{slot}[, \quad U = U(x(s_k)).$$

But now it may happen that occasionally a control task deadline is missed : in that case it is proposed to abort the current computation, hold the current value of the control signal for the next period and start a fresh computation with the next sensor value. In that case, the control signal $U(x(s_k))$ is hold for one extra

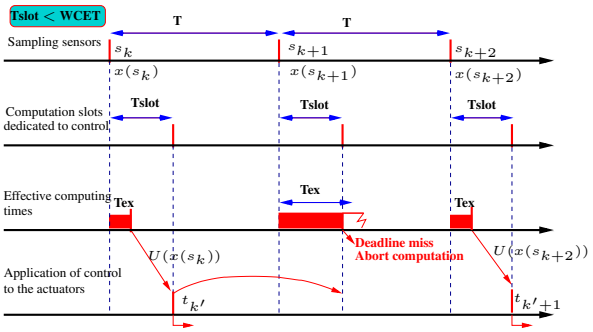


Figure 7: Abort scheduling pattern

period, i.e. if the deadline miss occur at time $s_k + T_{slot}$:

$$\forall t \in [s_k + T_{slot}, s_{k+2} + T_{slot}[, \quad U = U(x(s_k)),$$

and for N consecutive deadline misses and data loss:

$$\forall t \in [s_k + T_{slot}, s_{k+N} + T_{slot}[, \quad U = U(x(s_k)).$$

Then, the control input can be asynchronous since the difference between two sampling instant $t_{k'+1} - t_{k'}$ is time-varying but bounded by T and NT.

2 Abort' Again, a time slot $T_{slot} < WCET$ is allocated to the control task, but the system's period is now also reduced by the same value, while the time remaining for computing other activities remains T_{others} as in the initial scheme. In that case control improvement is expected from both the latency and sampling period reduction (Figure 8).

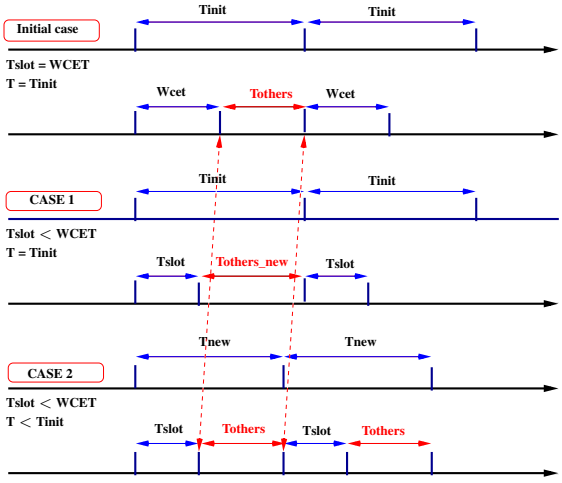


Figure 8: Abort and Abort' scheduling cases

From the computing point of view, these two schemes might be not as simple to implement as it seems at first glance. Indeed, in case of deadline miss it is necessary to

instantaneously abort the running instance of the control task and cleanly restart a new instance with the new sensors data, and with cleanly reinitialized internal variables, filters and other related data structures. This scheduling pattern may be difficult to implement, e.g. in Posix pthreads written in C where a thread's function (used to implement a periodic control task) cannot be easily aborted and immediately restarted. The following pattern is in such cases easier to carry out.

3 Skip For the nominal case this pattern is similar to Case 1, where the period of the control task is kept equal to the initial case. However, in case of deadline miss, rather than aborting the running instance, the running task continues its computation until reaching its completion. The execution of the task which begun during slot T_{slot_k} continues during the following slot allocated for this task ($T_{slot_{k+1}}$), and possibly on the following allocated slots $T_{slot_{k+n}}$, until completion. The last computed control signal is hold on the actuator input until a new one has been produced and sent to the actuators, as in Figure 9 where $U(x(s_k)$ is hold until $s_{k+2} + T_{slot}$ due to the deadline miss during slot $T_{slotk+1}$. Note that here the control signal sent at time $s_{k+2} + T_{slot}$ is $U(x(s_{k+1}))$ computed using the sensor's value s_{k+1} received one period and one slot before.

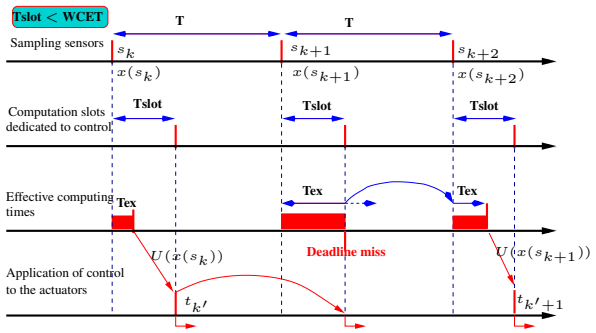


Figure 9: Skip scheduling pattern

4 Spare Note that some spare computing time can be reserved inside the T_{others} slots. Usually this spare time is used for housekeeping activities, but it could also be used to handle the extra computation cycles needed to fulfill the computation of an over-running control task which slides to the spare slots (Figure 10). In that case, if the spare slot is long enough to finish the running computation, the control signal can be sent either just at the end of the spare slot (to minimize the I/O latency) or at the end of the period (to minimize the output jitter). The remaining computations may slide until the next available spare slots until completion (or In both cases the nominal scheduling pattern can be recovered at the next received sensor signal.

5 Skip' Combining the previous cases 2 and 3, it is proposed to reduce the control period according to the reduction of T_{slot} (as in the **Abort'** case), and to adopt the **Skip** overrun handler in case of deadline miss.

7-> Just-in-time In all the previous cases, the newly computed control signal is sent to the actuator waiting the end of the allocated T_{slot} to allow for minimal jitter and equidistant sampling at the actuator's input. However, the control signal

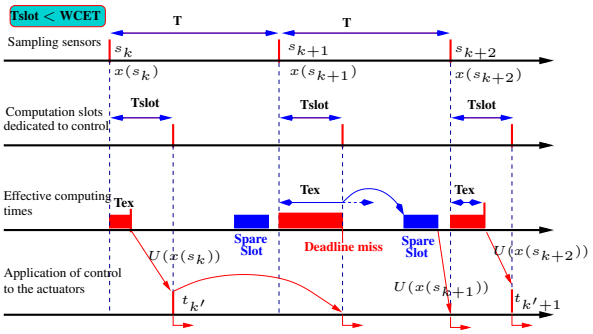


Figure 10: Slipping on a spare slot

can also be sent to the actuator immediately after the computation completion. In that case it is expected (e.g. following Buttazzo and Cervin [2007]) that the disturbance due to the output jitter can be compensated by the average lowered value of the I/O latency, therefore finally increasing the control performance. This **Just-in-time** control sending strategy can be associated with any of the previously described scheduling patterns and overrun handlers.

...

Indeed the combination of possible scheduling schemes (including fully asynchronous or event-based control laws) with various overrun handlers is potentially very large. The right combination is the result of a trade-off to be evaluated for each case study between control algorithms, controlled plants, execution resources, operating systems capabilities and certification constraints...

6 Conclusion

The hard deadline of a critical control tasks is usually based on the knowledge of the WCET. This knowledge presupposes the existence of calculation methods which complexity are growing with modern processors, leading to conservative evaluations due to unpredictability and increasing spare margins. Anyway, even if an evaluation of the WCET has been found, execution times are variable and rarely reach the worst case value. Hence a control design and implementation based on a static schedule of actions, whose rate is based on the estimated WCET and the assumption that all timing deadlines are hard, inevitably lead to an under- utilization of the computing resources. However, the knowledge of both the plant's characteristics and feedback control system capabilities can be used to better understand and model the relations between the control performance and the control tasks scheduling parameters. In particular recognizing that a feedback controller is robust w.r.t. timing uncertainties and deviations from a rigid timing pattern allows to slacken the usually considered hard real- time constraints. As solving a too general control/scheduling co-design problem seems to be out of range of feasible solutions, restricted assumptions sets can be used to investigate more specific case studies. It is expected that well formalized cost functions to model the behavior of a real- time controller may lead to design effective and convincing flexible scheduling schemes even for critical control systems.

References

T.F. Abdelzaher, E.M. Arkins, and K.G. Shin. Qos negotiation in real-time systems and its application to automated flight control, 1997.

M. Alden and F. Crusca. Quadratic cost function design for linear optimal control systems. In *IEEE Region 10 conference*, Tencon 92, Melbourne Australia, 1992.

Patrick Jocelyn Andrianiaina, Alexandre Seuret, and Daniel Simon. Robust control under weakened real-time constraints. In *CDC/ECC 2011*, Orlando, United States, December 2011. URL `http://hal.inria.fr/hal-00640403/en`.

K.-E. Årzén, A. Cervin, J. Eker, and L. Sha. An introduction to control and scheduling co-design. In *39th IEEE Conference on Decision and Control*, Sydney, Australia, December 2000.

M. Ben Gaïd, D. Simon, and O. Sename. A Design Methodology for Weakly-Hard Real-Time Control. In *17th IFAC World Congress*, Seoul, Korea, 2008.

G. Bernat, A. Burns, and A. Llamosí. Weakly hard real-time systems. *IEEE Trans. on Computers*, 50(4):308–321, 2001.

G. Bernat, A. Colin, and S-M. Petters. WCET analysis of probabilistic hard real-time systems. In *23rd IEEE Real-Time Systems Symposium RTSS'02*, Washington DC, USA, 2002.

D. Bertrand, S. Faucou, and Y. Trinquet. An analysis of the autosar os timing protection mechanism. In *14th IEEE international conference on Emerging technologies & factory automation ETFA'09*, Palma de Mallorca, Spain, 2009.

C. Burguière and Ch. Rochange. History-based schemes and implicit path enumeration. In *6th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, Dagstuhl, Germany, 2006.

G. Buttazzo and A. Cervin. Comparative assessment and evaluation of jitter control methods. In *Proc. 15th International Conference on Real-Time and Network Systems*, Nancy, France, March 2007.

A. Cervin. Using Jitterbug to derive control loop timing requirements. In *CERTS'03 – Co-Design of Embedded Real-Time Systems Workshop*, Porto, Portugal, July 2003.

A. Cervin. Towards the integration of control and real-time scheduling design. Licentiate thesis tfrt-3226, Dpt. of Automatic Control, Lund University, Sweden, May 2000.

A. Cervin and B. Lincoln. Jitterbug 1.1—Reference manual. Technical Report ISRN LUTFD2/TFRT--7604--SE, Department of Automatic Control, Lund Institute of Technology, Sweden, January 2003.

A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1–2):25–53, July 2002.

A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo. The jitter margin and its application in the design of real-time control systems. In *10th International Conference on Real-Time and Embedded Computing Systems and Applications*, Göteborg, Sweden, August 2004.

A. Cervin, K-E. Årzén, D. Henriksson, M. Lluesma Camps, P. Balbastre, I. Ripoll, and A. Crespo. Control loop timing analysis using TrueTime and Jitterbug. In *2006 IEEE Computer Aided Control Systems Design Symposium*, October 2006.

Anton Cervin. Analysis of overrun strategies in periodic control tasks. In *Proc. 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

J. Deverge and I. Puaut. Safe measurement-based wcet estimation. In *5th International Workshop on worst-case execution time analysis*, Palma de Mallorca, Spain, july 2005.

R. Dorf, M. Farren, and C. Phillips. Adaptive sampling frequency for sampled-data control systems. *IEEE Trans. on Automatic Control*, 7(1):38–47, 1962.

J. Eker, P. Hagander, and K.-E. Årzén. A feedback scheduler for real-time controller tasks. *Control Engineering Practice*, 8(12):1369–1378, January 2000.

Ph. Goupil and A. Marcos. *Fault Tolerant Flight Control*, chapter ch. 19: Industrial Review, pages 521–536. Number 399 in LNCIS. Springer-Verlag, 2010.

A. Hamann, R. Racu, and R. Ernst. A formal approach to robustness maximization of complex heterogeneous embedded systems. In *4th international conference on Hardware/software codesign and system synthesis CODES+ISSS'06*, 2006.

T. Henzinger and J. Sifakis. The embedded systems design challenge. In *14th International Symposium on Formal Methods (FM)*, number 4085 in LNCS, Hamilton, Canada, August 2006. Springer.

T. C. Hsia. Comparisons of adaptive sampling control laws. *IEEE Trans. on Automatic Control*, 17(6):830–831, 1972.

T. C. Hsia. Analytic design of adaptive sampling control law in sampled-data systems. *IEEE Trans. on Automatic Control*, 19(1):39–42, 1974.

R. Kirner and P. Puschner. Classification of WCET analysis techniques. In *8th IEEE International Symposium on Object-oriented Real-time distributed Computing ISORC'05*, pages 190–199, Seattle, Washington, May 2005.

R. Kirner, R. Lang, P. Puschner, and C. Temple. Integrating WCET analysis into a Matlab/Simulink simulation model. In *16th IFAC Workshop on Distributed Computer Control Systems*, Sydney, Australia, 2000.

B. Lisper. Fully automatic, parametric worst-case execution time analysis. In *Workshop on Worst-Case Execution Time (WCET) Analysis*, pages 3–0531, 2003.

S. Louise. *Calcul de majorants sûrs de temps d'exécution au pire pour des tâches d'application Temps-réel critique pour des sytèmes disposant de caches mémoire*. PhD thesis, Paris XI Orsay, 2002.

P. Puschner and M. Schoeberl. On composable system timing, task timing, and WCET analysis. In *8th International Workshop on Worst-Case Execution Time (WCET) Analysis*, Prague, Czech Republic, July 2008.

D. Robert, O. Sename, and D. Simon. An H_∞/LPV design for sampling varying controllers : experimentation with a T inverted pendulum. *IEEE Trans. on Control Systems Technology*, 18(3):741–749, May 2010.

Ch. Rochange, editor. *7th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, Pisa, Italy, jul 2007.

D. Seto, J.-P. Lehoczky, L. Sha, and K.-G. Shin. On task schedulability in realtime control systems. 17^{th} *IEEE Realtime Systems Symposium*, 1996.

Alexandre Seuret. Stability analysis of networked control systems with asynchronous sampling and input delay. In *2011 American Control Conference - ACC 2011*, page 6, San Francisco, Californie, United States, June 2011. URL `http://hal.inria.fr/hal-00578771/en`.

L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real-time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2–3):101–155, November 2004.

K.-G. Shin and H. Kim. Derivation and application of hard deadlines for realtime control systems. *IEEE Trans. on Systems, Man and Cybernetics*, 22(6), november/december 1992.

K.-G. Shin, C.M. Krishna, and Y.-H. Lee. A unified method for evaluating realtime computer controllers and its application. *IEEE Trans. on Automatic Control*, 30(4), april 1985.

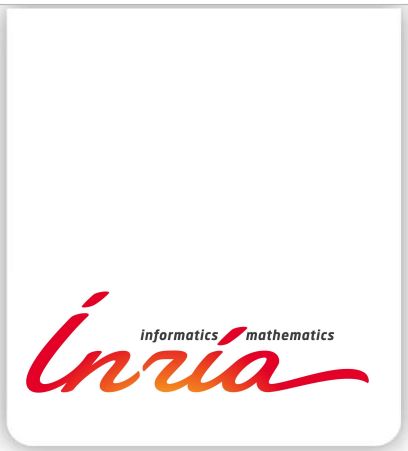
K.G. Shin and C.L. Meissner. Adaptation and graceful degradation of control system performance by task reallocation and period adjustment. In *11th Euromicro Conference on Real-Time Systems*, York, England, 1999.

D. Simon, A. Seuret, P. Hokayem, J. Lygeros, and E. Camacho. State of the art in control/computing co-design. Technical Report Public Deliverable D04-01, FeedNetBack IST FP7 project, 2009. URL `http://feednetback.eu/`.

M.J. Smith. An evaluation of adaptive sampling. *IEEE Trans. on Automatic Control*, 16(3):282–284, 1971.

J. Souyris, E. Le Pavec, G. Himbert, V. Jégu, and G. Borios. Computing the worst case execution time of an avionics program by abstract interpretation. In *5th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*, pages 21–24, 2005.

F. Xia and Y. Sun. Control-scheduling codesign: A perspective on integrating control and computing. *Dynamics of Continuous, Discrete and Impulsive Systems - Series B*, Special Issue on ICSCA 06:1352–1358, 2006.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399